Prefix reminder: nano = 10^{-9}, pico=10^{-12}, femto=10^{-15}

Problems 1-2 refer to the schematic diagram of the recirculating latch shown below.

1. **Sequential Circuit Timing**  
   20 Points

   (a) Fill in the timing diagram below to show how the recirculating latch responds to changes in its inputs. Be sure to show delays in your timing diagram. Note: assume that the values of D, DT, /Q, and Q are initially unknown. Mark unknown values by cross-hatching that part of the timing diagram.
(b) As described in the text, every latch has a setup time. Briefly describe the meaning of setup time and describe the factors which determine setup time in this circuit.

2. Sequential Circuit Layout

20 Points

Sketch a layout for the recirculating latch in which Vdd and Gnd feed from left to right, clock signals feed from top to bottom, D input is on the left, and Q output is on the right.

Problems 3-6 refer to the VHDL description shown attached to the end of this exam. This description contains two modules (a sequential and a finite state machine) that are connected together in a third structural module. Study the VHDL description carefully before answering the following questions.

3. VHDL Sequential Circuit “SQ” 15 Points

(a) What common sequential logic building block is implemented by this circuit?

(b) Under what conditions are the rdy1 and rdy2 outputs true?
4. **VHDL Finite State Machine**  
15 Points

Draw a State Diagram or ASM diagram that describes the operation of the FSM module.

5. **VHDL Structural Description**  
15 Points

Draw a block diagram of the sysbox structural description showing the inputs and outputs of each module and how they are connected to each other and the outside world.

6. **VHDL Control Unit / Data Unit Operation**  
15 Points

The FSM and SQ modules respectively act as control unit and data unit in this example. Fill in the timing diagram below to show the two modules interact together to determine the overall operation of the circuit. Include FSM state and internal signals rdy1 and rdy2.
VHDL Description Part 1: Sequential Circuit

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity SQ is
port(
   dout: out std_logic_vector(2 downto 0);
   rdy1, rdy2: out std_logic;
   reset, clk : in std_logic);
end;

architecture behavioral of SQ is

signal present_out,next_out : std_logic_vector(2 downto 0);

begin

comb_process: process(present_out, reset, clk)
begin
   dout <= present_out;
   rdy1 <= '0';
   rdy2 <= '0';
   if (reset = '1') then next_out <= "000";
   else
      if (present_out = "101") then
         rdy1 <= '1';
         next_out <= present_out + 1;
      elsif (present_out = "111") then
         rdy2 <= '1';
         next_out <= "000";
      else
         next_out <= present_out + 1;
      end if;
   end if;
end process comb_process;

clk_process: process
begin
   wait until (clk'event and clk='1');
   present_out <= next_out;
end process clk_process;

end behavioral;
Library IEEE;
use IEEE.std_logic_1164.all;

entity fsm is
port( start, rdy1, rdy2, clk: in std_logic;
     reset, flag: out std_logic);
end fsm;

architecture behavioral of fsm is
type state_type is (S0, S1, S2);
signal current_state, next_state : state_type := S0;
begin

comb_process: process(current_state, start, rdy1, rdy2)
begin
  flag <= '0';
  reset <= '0';
  if (start = '1') then next_state <= S0;
  else
    case current_state is
      when S0 =>
        reset <= '1';
        next_state <= S1;
      when S1 =>
        flag <= '1';
        if (rdy1 = '1') then next_state <= S2;
        else next_state <= S1;
        end if;
      when S2 =>
        flag <= '0';
        if (rdy2 = '1') then next_state <= S1;
        else next_state <= S2;
        end if;
    end case;
  end if;
end process comb_process;

clk_process: process
begin
  wait until (clk'event and clk='1');
  current_state <= next_state;
end process clk_process;
end behavioral;
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity sysbox is
  port(clk, start : in std_logic;
       flag : out std_logic;
       dout : out std_logic_vector(2 downto 0));
end sysbox;

architecture structural of sysbox is

  component FSM
    port( start, rdy1, rdy2, clk: in std_logic;
          reset, flag: out std_logic);
  end component;

  component SQ
    port(
      dout: out std_logic_vector(2 downto 0);
      rdy1, rdy2: out std_logic;
      reset, clk : in std_logic );
  end component;

  for all: FSM use entity work.FSM(behavioral);
  for all: SQ use entity work.SQ(behavioral);

  signal reset, rdy1, rdy2 : std_logic;

begin

  F1: FSM port map(start=>start, clk=>clk, rdy1=>rdy1, rdy2=>rdy2,
                   reset=>reset, flag=>flag);

  S1: SQ port map(dout=>dout, rdy1=>rdy1, rdy2=>rdy2,
                  reset=>reset, clk=>clk);

end structural;